

JavaTM magazine

By and for the Java community 

Seize the Cloud

Develop, deploy, and manage your applications in the cloud with Java EE and Oracle Java Cloud Service

38

HOW DEVELOPERS ARE
USING THE CLOUD

42

WHAT TO LOOK FOR IN A
CLOUD SERVICE VENDOR



20

2013 DUKE'S
CHOICE AWARDS



COMMUNITY

03

From the Editor

05

Java Nation

News, people, and events

16

JCP Executive Series

Q&A with Gemalto M2M

Thomas Lampart and Florian Denzin on M2M, Java, and the JCP

JAVA TECH

28

Java Architect

Java 8: Lambdas

Part 2 in Ted Neward's series on getting to know lambda expressions

45

Enterprise Java

Building Rich Client Applications with JSR 356: Java API for WebSocket

Build a simple chat application with the JSR 356 server API and an HTML5 client.

52

Rich Client

JavaFX Data Binding for Enterprise Applications

Part 2 in Adam Bien's series on data flow in the JavaFX API

58

Mobile and Embedded

Enter a Brave New World—Embedded Java at Your Fingertips

Vikram Goyal presents the basics of embedded Java.

61

Fix This

Take our class initialization code challenge!



20

2013 DUKE'S CHOICE AWARDS

Meet the 11 winners of this year's awards, which honor innovation in Java.

36

CLOUD: CHALLENGE AND OPPORTUNITY

Oracle's Cameron Purdy weighs in on how developers can take advantage of cloud computing today.

42

Enterprise Java HANDS ON WITH ORACLE JAVA CLOUD SERVICE

Get an introduction to Oracle's platform-as-a-service Java offering.

EDITORIAL

Editor in Chief

[Caroline Kvitka](#)

Community Editors

Cassandra Clark, Sonya Barry,
Yolande Poirier

Java in Action Editor

Michelle Kovac

Technology Editors

Janice Heiss, Tori Wieldt

Contributing Writer

Kevin Farnham

Contributing Editors

Claire Breen, Blair Campbell, Karen Perkins

DESIGN

Senior Creative Director

Francisco G Delgadillo

Senior Design Director

Suemi Lam

Design Director

Richard Merchán

Contributing Designers

Jaime Ferrand, Nicholas Pavkovic,
Arianna Pucherelli

Production Designers

Sheila Brennan, Kathy Cygnarowicz

PUBLISHING

Vice President

Jeff Spicer

Publisher

[Jennifer Hamilton](#) +1.650.506.3794

Audience Development and

Operations Director

[Karin Kinnear](#) +1.650.506.1985

ADVERTISING SALES

Associate Publisher

[Kyle Walkenhorst](#) +1.323.340.8585

Northwest and Central U.S.

[Tom Cometa](#) +1.510.339.2403

Southwest U.S. and LAD

[Shaun Mehr](#) +1.949.923.1660

Northeast U.S. and EMEA/APAC

[Mark Makinney](#) +1.805.709.4745

Advertising Sales Assistant

[Cindy Elhaj](#) +1.626.396.9400 x 201

Mailing-List Rentals

Contact your sales representative.

RESOURCES

Oracle Products

+1.800.367.8674 (U.S./Canada)

Oracle Services

+1.888.283.0591 (U.S.)

Oracle Press Books

[oraclepressbooks.com](#)

ARTICLE SUBMISSION

If you are interested in submitting an article, please [e-mail the editors](#).

SUBSCRIPTION INFORMATION

Subscriptions are complimentary for qualified individuals who complete the [subscription form](#).

MAGAZINE CUSTOMER SERVICE

[java@halldata.com](#) **Phone** +1.847.763.9635

PRIVACY

Oracle Publishing allows sharing of its mailing list with selected third parties. If you prefer that your mailing address or e-mail address not be included in this program, contact [Customer Service](#).

Copyright © 2013, Oracle and/or its affiliates. All Rights Reserved. No part of this publication may be reprinted or otherwise reproduced without permission from the editors. *JAVA MAGAZINE* IS PROVIDED ON AN "AS IS" BASIS. ORACLE EXPRESSLY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS OR IMPLIED. IN NO EVENT SHALL ORACLE BE LIABLE FOR ANY DAMAGES OF ANY KIND ARISING FROM YOUR USE OF OR RELIANCE ON ANY INFORMATION PROVIDED HEREIN. The information is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle. Oracle and Java are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

Java Magazine is published bimonthly with a free subscription price by Oracle, 500 Oracle Parkway, MS OPL-3C, Redwood City, CA 94065-1600.

Digital Publishing by Texterity

COMMUNITY

JAVA IN ACTION

JAVA TECH

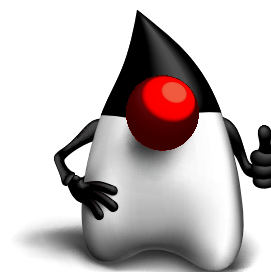
ABOUT US



02

Get Java EE Expertise

—Oracle University—



- ✓ New Java EE 7 Training
- ✓ Engineering-Developed Courses
- ✓ Aligned with Java EE Certifications
- ✓ Experienced Java Instructors

Register Now

ORACLE®

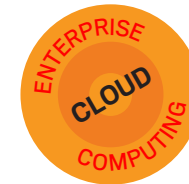
//from the editor /



T

he IT forecast is for clouds. Public clouds and private clouds are here to stay. In this issue, we look at what the cloud means for Java developers and give you the tools to seize the opportunity that cloud presents.

First, Oracle's Cameron Purdy weighs in on why developers need to change the way they think, in the interview "[Cloud: Challenge and Opportunity](#)." Applications must be built so that they can dynamically scale out and dynamically balance the load across multiple servers, he says, adding that developers must also explicitly design the ability for applications to scale as close to linearly as possible. Purdy also discusses why developers should pay attention to platform as a service. Plus: developer, Java Champion, and Netherlands JUG leader [Bert Ertman](#) tells us how he is using the cloud today.




Next, Java Champion Harshad Oak helps you get started with "[Hands On with Oracle Java Cloud Service](#)." While it was once unheard of to suggest that a Java EE application could run in a shared environment, says Oak, it is now actually fashionable to do so.

We also bring you the 11 winners of the [Duke's Choice Awards](#) (now in its 11th year). This year's winners of the award, which honors innovation in Java development, range from the serious and cerebral to the practical, amusing, and entertaining. The winners are pushing the frontiers of technology by simulating the human brain and musculoskeletal system; providing guidance to cars, satellites, and robotic fish; training tomorrow's Java programmers; making Java applications more secure; and building communities. We are honored to recognize them here.


How are you using the cloud? How are you innovating with Java? [Let us know](#).

Caroline Kvitka, Editor in Chief

PHOTOGRAPH BY BOB ADLER



MAKE THE
FUTURE
JAVA




FIND YOUR JUG HERE

One of the most elevating things in the world is to build up a community where you can hang out with your geek friends, educate each other, create values, and give experience to your members.

Csaba Toth
Nashville, TN Java Users' Group (NJUG)

[LEARN MORE](#)



ORACLE®

//send us your feedback /

We'll review all suggestions for future improvements. Depending on volume, some messages may not get a direct reply.



COMMUNITY

JAVA IN ACTION

JAVA TECH

ABOUT US

O

f

Twitter icon

java.net

blog

03

CMS or Portal? Choosing the Right Technology

dotCMS CTO Will Ezell takes a look at trends driving change for content management

Q> Why do companies have trouble deciding between these systems?

There's a fair amount of overlap between a CMS and a portal—many portal tools contain rudimentary content management tools.

But while companies may be able to use a portal in place of a CMS system, getting it to look and act the way you like might be difficult. You might end up trying to hammer a square peg into a round hole.

How so?

Companies often choose to use portals for their document libraries or content-rich intranets. The issue is that portals are really just frameworks intended to provide an aggregated view into multiple enterprise applications based on a user's profile. Think of the old My Yahoo pages, which contained a lot of customizable, lightweight widgets—it turns out that the widgets were too lightweight and users inevitably would leave for the fuller experience of the native sites.

The real problem is that today's business users expect any site—web, mobile, even intranets—to look and interact in ways that portals weren't designed for.



Will Ezell, CTO, dotCMS

So it's a question of function not meeting need?

Portals are good at what they are designed for, but can be cumbersome to build in. They require your sites to function in a specific way, which makes it hard on your users. And these days, users are used to dealing with websites and apps that are flexible, intuitive, and very easy to use. Turns out, your workforce wants their work-related web experiences to be just as simple and easy as their consumer experiences.

So CMS systems like dotCMS are built to address ease of use?

Absolutely. If your team has ever wrestled with the look, feel or interactivity of your portal-driven site or intranet, you should consider dotCMS. DotCMS is designed from the ground up to manage and seamlessly deliver role-based content, sites, mobile sites, intranets and applications without impacting interactivity or imposing rigid look-and-feel requirements. It moves content and user experience to the center of the equation, where it belongs. We basically provide the development tools and get out of the way.

How do Java developers react to working with dotCMS?

When J2EE developers hear from a business unit, "I need ABC web site that does XYZ," they instinctively reach for a portal framework.

It turns out that a CMS might offer an easier, more flexible platform for delivering such sites and apps. DotCMS is based on familiar open standards such as OSGi, CMIS, Spring, Struts, Hibernate, Velocity and Elasticsearch, which also means that developers can take our code base and customize it to do whatever they need.

Our UI is very straightforward and easy to comprehend; we tried to make dotCMS as approachable and simple as possible. Plus, our use of loosely coupled RESTful APIs means that you can read/write content and apps to any 3rd-party web-based system, be it php, .NET or J2EE. It's just much easier to respond to changes and manage any given site or content.

And this may be the best part: it's a fraction of the price of most portal solutions in this space.

For more information visit dotCMS.com





Top: Teens get hands-on practice using Alice at the Make the Future Java Summer Workshop. Bottom: David Culyba of Carnegie Mellon University describes the programming process.

PHOTOGRAPHS BY DON SLATER



JAVA'S FUTURE IS BRIGHT

1,000 San Francisco Bay Area teenagers learned how to program in Java at the **Make the Future Java Summer Workshop**, held July 30–August 1 at Oracle headquarters in Redwood Shores, California. Oracle Academy staff members and Carnegie Mellon University (CMU) computer science professors helped participants learn Java using the award-winning *Alice* visual 3-D educational tool, which uses a drag-and-drop interface to create animations. On August 1, students attended a *Minecraft* workshop, where they learned how to extend the popular online game *Minecraft* by using Java to write modifications.



Nick Rocha, a 13-year-old programmer from San Antonio, Texas, presented a session to all attendees called *Inspired By Alice*. Rocha, who has a passion for computer-based and

video games, learned to program a year ago through a Learning to Program with Alice course at Homeschool Arts & Academics Class Day. He was instantly hooked on Alice. "If you like games, 3-D animation, and being creative, I bet you'll like Alice," Rocha told the audience. He demoed several of the animations and games that he has built using Alice. "Alice can be tough," he explained, "but it's both fun and rewarding."

Attendees also heard from **David Culyba**, one of the primary Alice developers at CMU in Pittsburgh, Pennsylvania. Culyba, who worked on Alice as an undergraduate at CMU, worked in the industry as a game developer before joining the Alice team, of which Oracle is a long-time supporter. Culyba focuses on how to make programming less frustrating and more fun. When the audience was asked if he had succeeded, based on their day's experience

//java nation / javaone /

JAVAONE SHANGHAI

JavaOne Shanghai was held July 22–25 at the Shanghai Expo Center. Here are a few highlights:

Geeks ride. Now a tradition around the world, the Geek Bike Ride preceded JavaOne. In matching bike jerseys (pictured above), more than 50 riders enjoyed a ride near the Huangpu River. In deference to more than 90-degree heat and high humidity, the group stopped for refreshments halfway through the ride. It was great fun and camaraderie.

Java user group (JUG) meeting. On Sunday, the [Green Tea JUG](#) took advantage of the speakers coming to China for JavaOne. More than 150 developers heard speakers from Oracle, IBM, and Alibaba on topics ranging from Java EE new features to OpenJDK best practices. JUG leader **Leo Yu** said, “It was a great event!”

Embedded discussion. A panel discussion on “Embedded to the Enterprise” was held at the Asia Pacific Oracle User Group Summit. The discussion focused on the instrumental role of Java EE for the Internet of Things and the cloud going forward.

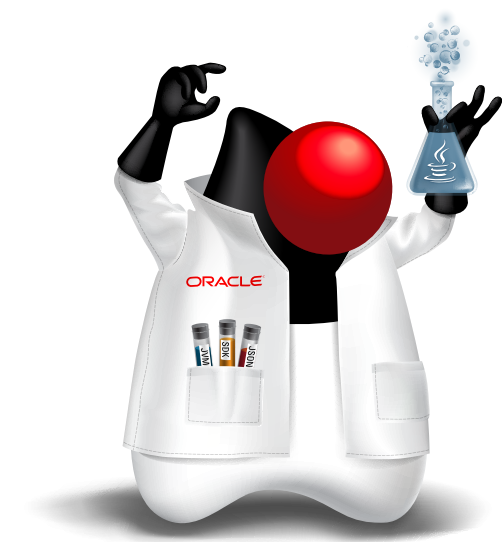
Java EE 7. Just released, Java EE 7 was a hot topic at the technical keynote. Java EE 7 content included a hands-on lab and sessions on the platform, WebSocket, and batch processing. An Oracle Java Cloud Service hands-on lab was also offered.



Alex Mironenko shows demos of embedded Java in action.



Simon Ritter goes head to head with Tori Wieldt at JavaOne Shanghai.



Meet Scientist Duke

Each year, Oracle releases a personality for Duke, the official mascot of Java technology.

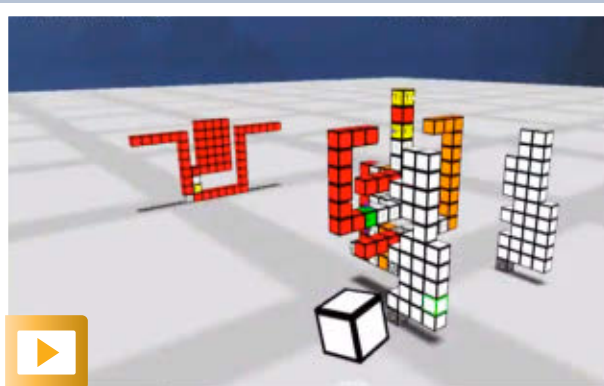
The latest is Scientist Duke. Much like a genius developer would with Java technologies, Scientist Duke is engaged in developing, mixing, and testing the latest ingredients to create the greatest formula/product/innovation ever. Last year, Adventure Duke was off on a mission to find the latest Java wonders around the world. 2011 unveiled Future Tech Duke, and 2010 brought us Surfing Duke.

jMonkeyEngine Combines Gaming and Community

If you like game development and open source, and you want to participate in an active Java community, check out [jMonkeyEngine](#). It is an all-Java game engine library written especially for game developers who want to create 3-D games using modern technology standards. The [core team](#) consists of nine people, led by architect **Kirill Vainer** and community builder **Erlend Sogge Heggen**.

One of the unusual features of the project is the completeness of the software documentation. For example, the recently released Version 3 includes the *jMonkeyEngine 3.0 Beginner's Guide*, a book written by jMonkeyEngine documentation specialist **Ruth Kusterer**.

The [jMonkeyEngine community site](#) includes a blog, documentation, forums, a download section, and ways to contact the project leaders. There are many ways to get involved: the [jMonkeyEngine Contributor's Handbook](#) describes roles for modelers, web developers, technical writers, programmers, and anyone else who'd like to help.

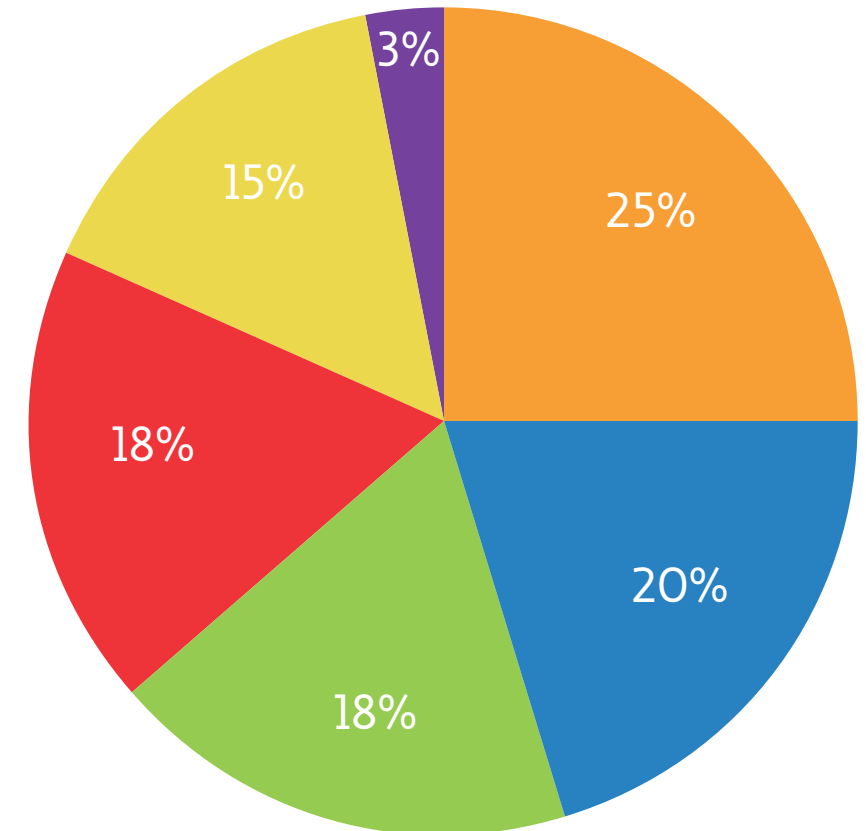


Get a look at what you can do with jMonkeyEngine.

JAVA.NET POLL

THE CLOUD: MORE THAN BUZZ

The results of a recent [Java.net poll](#) suggest that developers have widely varying views on the significance of the cloud, ranging from "It changes everything" to "It's just buzz." A total of 177 votes were cast during the two-week survey in response to the question "Does 'the Cloud' change anything?" Here are the final results:



25%
It's a pretty significant development.

20%
It changes everything: ultimately, it will eliminate most corporate data centers.

18%
It's too early to tell.

18%
It's not as important a development as many people think.

15%
No, it's just buzz.

3%
Other



EVENTS

Devovx NOVEMBER 11-15
ANTWERP, BELGIUM

Devoxx has been running for the last 12 years and is the largest Java conference in Europe. The five-day conference attracts world-class speakers and developers from 40 different European countries. Its program has a packed schedule with university sessions during the first two days and keynotes, sessions, hands-on labs, quickies, and birds-of-a-feather sessions over two days. Tracks include Java, Android, HTML5, and more.

PHOTOGRAPH BY GETTY IMAGES

Silicon Valley Code Camp

OCTOBER 5-6

LOS ALTOS HILLS, CALIFORNIA

At this free community event, developers learn from each other. In addition to programming training, the conference tackles topics such as software branding and legal issues around software.

CodeMotion

OCTOBER 17-19

MADRID, SPAIN

This conference is open to users of all languages and platforms. It offers full-day workshops during the first day, and keynotes, conference sessions, and hackathons in eight tracks over two additional days.

lazon

OCTOBER 22-23

ZURICH, SWITZERLAND

This international conference for the software community offers content on programming, frameworks, tools, platforms, and methods as well as about the professional development process for enterprises, the web, the cloud, mobile devices, and desktop applications.

JAX London

OCTOBER 28-30

LONDON, ENGLAND

JAX is about a new generation of web technologies and cloud-based services; sophisticated architectural

decisions for maximum flexibility and scalability; integration of big data technologies; and on the client side, the rise of powerful web apps and mobile post-PC devices.

Oredev 2013

NOVEMBER 4-8

MALMO, SWEDEN

The theme of the conference is “Year of the Arts.” Developers, testers, and project managers are invited to reflect on the art of the programming processes, languages, tools, and platforms. The organizers promise sessions and keynotes that are creative, provocative, and innovative.

W-]AX Munich

NOVEMBER 4-8

MUNICH, GERMANY

W-JAX is a leading conference that focuses on the Java platform, web, architecture, agile, and the cloud.

]-Fall

NOVEMBER 6

NIJKERK, THE NETHERLANDS

] - Fall is the annual Java developer conference organized by NLJUG, the Dutch Java user group. More than 40 technical presentations will be offered to the 1,200 expected attendees by local and international Java superstars. The conference marks the tenth anniversary of NLJUG and] - Fall.

Oracle Publishers Program

Oracle Publishers Program works with participating independent publishers to support the creation of high-quality technology books and related materials that focus on Oracle technologies. Independently published technology books are valuable in many ways, including raising awareness of Oracle products; making Oracle technology approachable for beginners; helping our customers be more successful; increasing product adoption; and generating buzz for new products and technologies. Oracle believes that publishers and authors are critical evangelists for Oracle, and that it is important to support the publication of Oracle technology books for the retail marketplace.

The Oracle Publishers Program provides support to member publishers by providing access, whenever possible, to product information, early product releases, and

Oracle experts, thus helping improve time to market for Oracle technology titles.

Publishers Program members and their nominated authors participate in our annual Oracle Publishers Seminar held in San Francisco and have the opportunity to attend Oracle's annual technology conferences—Oracle OpenWorld and JavaOne—which run at the same time as the seminar.

Book Promotions and Discount Channels

- Oracle Author Podcast Series
- *Oracle Magazine* Book Beat
- Java Nation
- Oracle Technology Network Member Discount
- Oracle Technology Network Member New Offers
- Oracle ACE Newsletter
- Oracle OpenWorld and JavaOne bookstore exposure and author signing sessions

Oracle Press

Your Destination for Java Expertise

Written by leading technology professionals, Oracle Press books offer the most complete, up-to-date coverage of Java available. Visit the JavaOne onsite bookstore to purchase these and other new Oracle Press books!

Don't miss these Oracle Press author sessions at JavaOne!

Ed Burns is a Senior Staff Engineer for Oracle where he chairs the JSR 344 (JSF 2.2) Expert Group. He is the leader of JavaServer Faces and has used Hudson extensively.

- JSF 2.2 New Features in Context [CON3294]
- JSF for Multitenant-Enabled Applications [CON3298]
- What's New in Portlet 3.0 and JSF 2.2 [CON7809]

Winston Prakash is the Eclipse Hudson project leader and an expert on its architecture and implementation.

- Hudson: The Little Heart of Big Enterprises [CON3025]

Danny Coward is a software architect for the Java Platform and was the Specification Lead for JSR 356 – Java WebSocket.

- JSR 356: Inside the Java WebSocket API [CON3436]

Coming soon—available for preorder

Java EE Applications on the Oracle Java Cloud
Harshad Oak

Java EE and HTML5 Enterprise Application Development
Geertjan Wielenga, Arun Gupta, John Brock

Mastering Lambdas: Java Programming in a Multicore World
Maurice Naftalin

Java: The Complete Reference, 9th Ed.
Herbert Schildt

Java: A Beginner's Guide, 6th Ed.
Herbert Schildt

Available in print and as eBooks.

Join the Oracle Press Community:
www.OraclePressBooks.com



Hudson Continuous Integration in Practice

*Ed Burns and
Winston Prakash*

Filled with best practices for implementing CI with Hudson, this book shows you how to streamline and stabilize each process in your development lifecycle.



Java WebSocket Programming

Danny Coward

Create, deploy, and secure dynamic enterprise server and client applications with the WebSocket protocol.



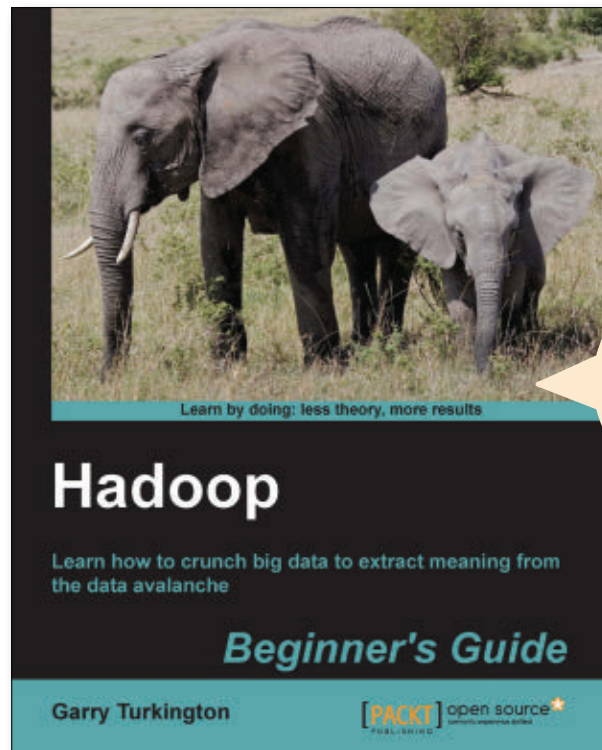
@OraclePress



OraclePress



Need to talk about the elephant in the room?



Claim this
\$29.99
eBook
for **FREE**
now

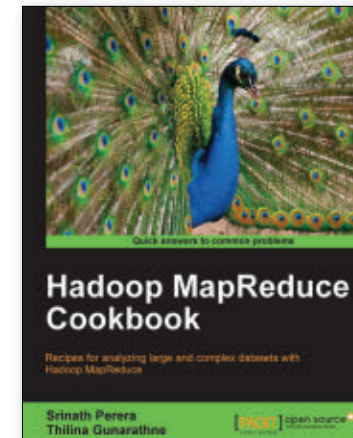
Claim your **FREE**
Hadoop Beginner's Guide eBook today at:
packtpub.com/javamag



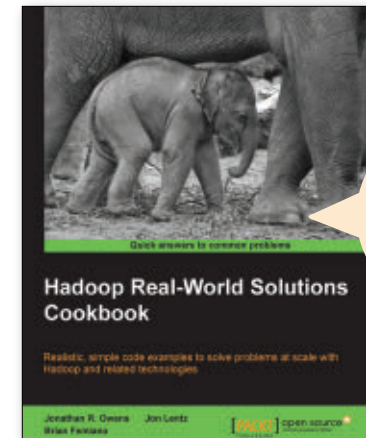
Offer expires November 30, 2013

PLUS save **30%** on this selection of
bestselling Hadoop titles at
packtpub.com/hadoopoffers for a limited time

Use code **JAVAONE13** to save 30% off the
eBook and print book prices for these titles

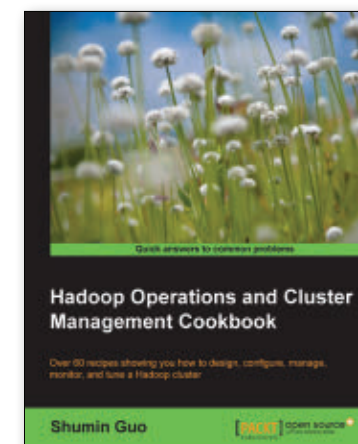


Hadoop MapReduce
Cookbook



Hadoop Real-World
Solutions Cookbook

eBooks
only
\$15.00 –
save **50%**



Hadoop Operations
and Cluster
Management Cookbook

Offer expires November 30, 2013

Be first to hear about the latest Hadoop and Java news and new title releases

Visit us: packtpub.com | Follow us: facebook.com/packtpub | twitter.com/packtpub



Save on the Books You Need to Design, Develop, and Deploy Secure Software



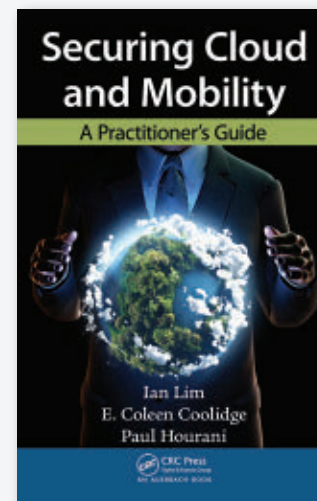
The Seven Qualities of Highly Secure Software

ISBN: 978-1-4398-1446-8, 160 pp.
\$49.95 / £31.99



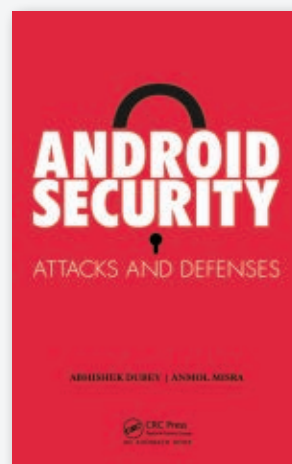
Secure Java For Web Application Development

ISBN: 978-1-4398-2351-4, 308 pp.
\$76.95 / £48.99



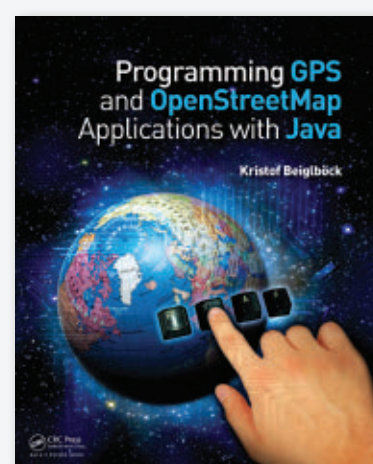
Securing Cloud and Mobility

ISBN: 978143985055-8, 228 pp.
\$79.95 / £49.99



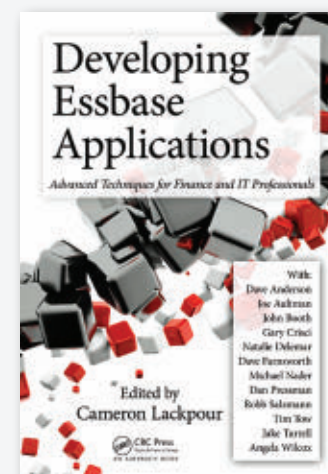
Android Security

ISBN: 978-1-4398-9646-4, 280 pp.
\$59.95 / £38.99



Programming GPS and OpenStreetMap Applications with Java

ISBN: 978-1-4665-0718-0, 248 pp.
\$59.95 / £38.99



Developing Essbase Applications

ISBN: 978-1-4665-5330-9, 445 pp.
\$69.95 / £44.99

Sign Up for our
Free Newsletters
and Connect with
CRC Press IT Books on
Facebook, Twitter,
and LinkedIn
to Keep the
Discounts Coming!



The final word in information
systems security



In-depth... insightful...for to-
day's technology leaders



Improving organizational
performance through IT



WWW.CRCPRESS.COM | CRC PRESS Taylor & Francis Group

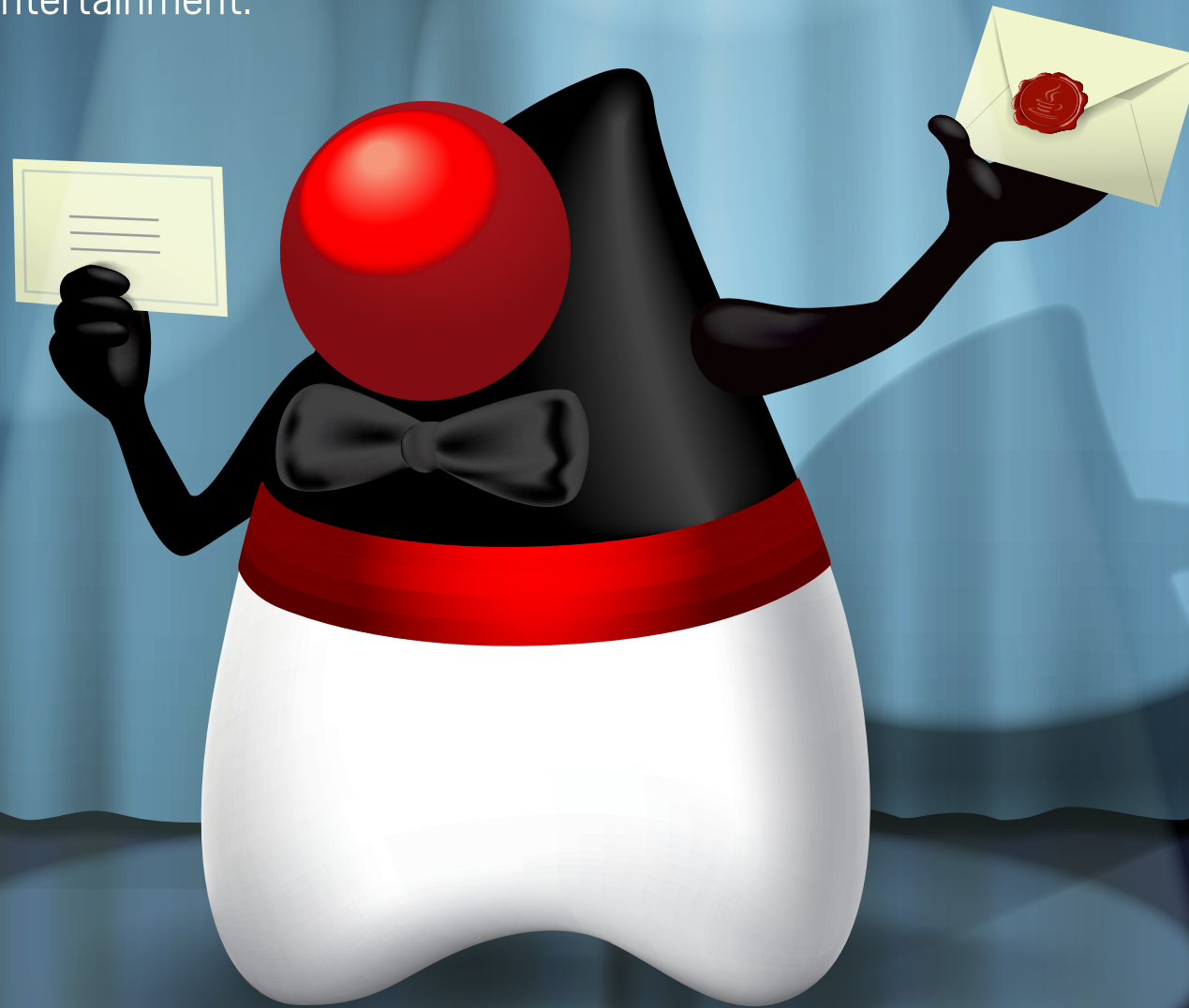
Order online and enter discount code **GVM24** to **SAVE 25%**

2013 DUKE'S CHOICE AWARDS

From the serious to the whimsical, the 2013 Duke's Choice Award winners include Java projects in medicine, technology, transportation, aeronautics, education, and entertainment.

BY PHILIP J. GILL

This year's Duke's Choice Award winners honoring innovation in Java development range from the serious and cerebral to the practical, amusing, and entertaining. These projects are pushing the frontiers of medicine and technology by simulating the human brain and musculoskeletal system; providing guidance to cars on the highway, satellites in space, and robotic fish under water; training tomorrow's Java programmers; making Java applications more secure; and building communities.



ART BY I-HUA CHEN

THIS YEAR'S WINNERS

(in alphabetical order
by organization name)

Contrast, [Contrast Security](#)

Devoux4Kids, [DEVOXX](#)

[The Dutch Java User Group](#)

ISIM, [ISBAK](#)

Bintray, [JFrog](#)

jCardSim, [Licel](#)

GEONS Ground Support
System, [National Aeronautics
and Space Administration](#)

OpenSim, [The National
Institutes of Health Center
for Biomedical Computation
and National Center for
Simulation in Rehabilitation
Research](#)

[openHAB](#)

Jessikommand, [Robotswim](#)

Neuroph, [University of
Belgrade's Faculty of
Organizational Sciences](#)



SERIOUS AND CEREBRAL

Musculoskeletal disorders and diseases such as rheumatoid arthritis and osteoporosis will affect one in two Americans during their lifetimes, and they are a leading cause of disability and healthcare costs, according to the publication *The Burden of Musculoskeletal Diseases in the United States* (American Academy of Orthopaedic Surgeons, 2011). To address this issue, the National Institutes of Health (NIH) funds various research initiatives, including the NIH Center for Biomedical Computation (known as Simbios) and the National Center for Simulation in Rehabilitation Research (NCSRR) at Stanford University, in Stanford, California.

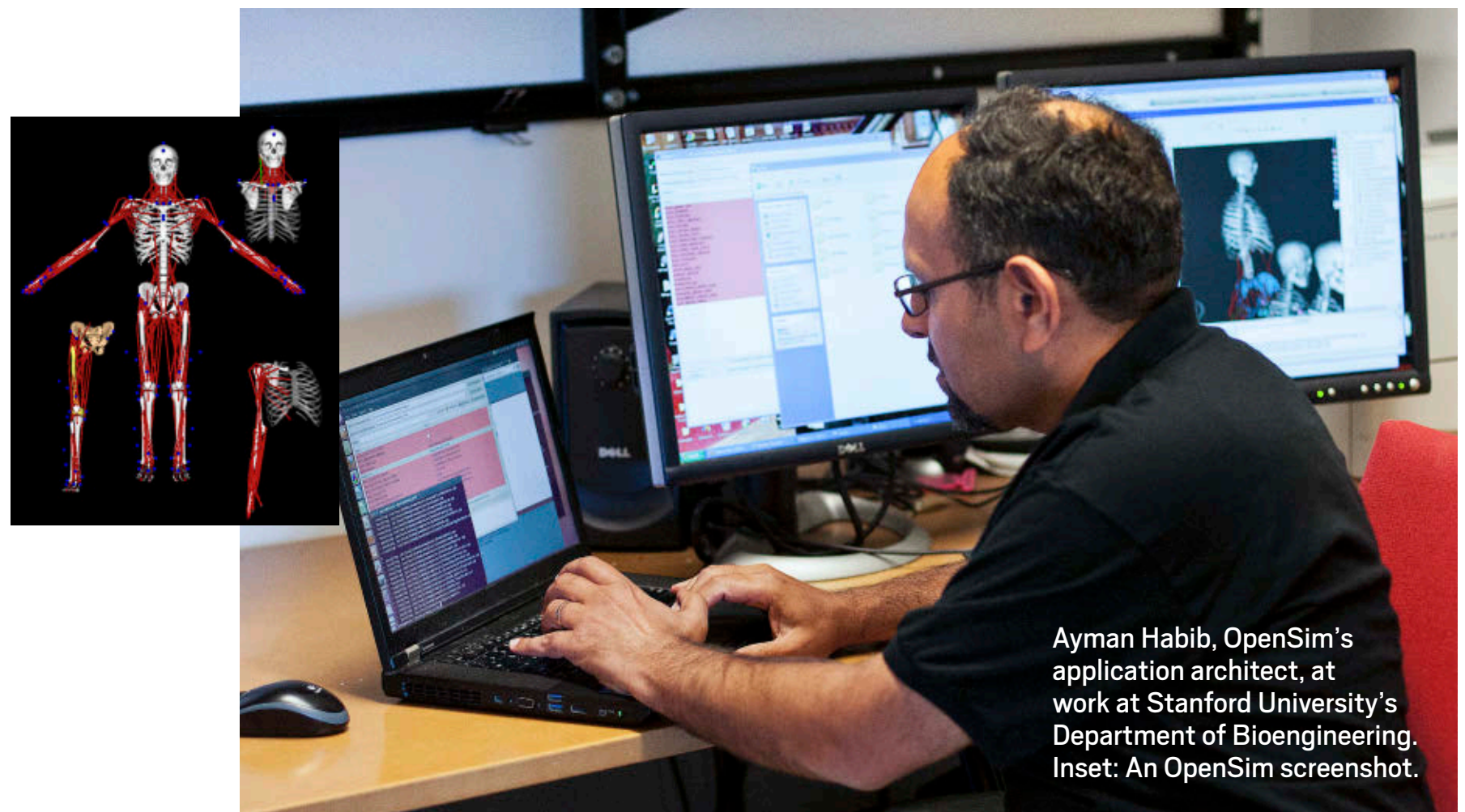
The team at Simbios and the NCSRR created **OpenSim**, an application for modeling the muscles,

PHOTOGRAPH BY BOB ADLER;
OPENSIM SCREENSHOT COURTESY
OF JEN HICKS AND KEVIN XU

DECIDING FACTOR
“Cross-platform
was key to our
selection of Java
as the GUI front
end to OpenSim.”
—Ayman Habib,
Application Architect,
OpenSim

joints, and bones that make up the body and simulating how humans move. This free tool enables researchers, therapists, students, and product designers to develop, analyze, simulate, and share information to find treatments—and, perhaps one day, cures—for a variety of musculoskeletal disorders and diseases. In addition, teams from DARPA’s *Warrior Web* effort are using the software to help design the next generation of “smart suits” for soldiers to reduce injury risk and fatigue.

“OpenSim is an open platform that can be easily shared because it is built on an open-system, open source approach that includes Java technology and the [NetBeans Platform](#),” explains Ayman Habib, OpenSim’s application architect and a member of the Neuromuscular Biomechanics Lab in Stanford University’s Department of Bioengineering. “OpenSim models are composed of components that can be shared,” Habib notes. If researchers develop a new type of muscle model, for example, they can write the



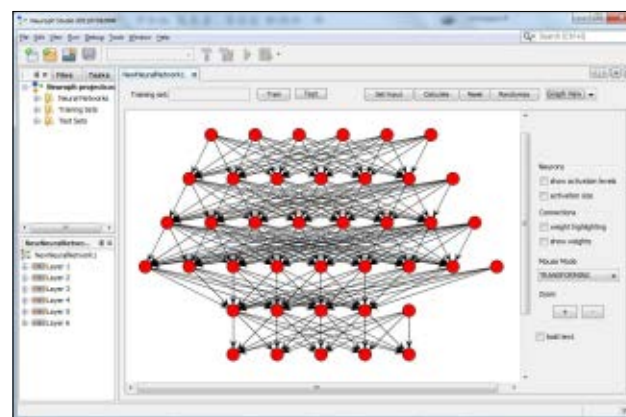
Ayman Habib, OpenSim’s application architect, at work at Stanford University’s Department of Bioengineering. Inset: An OpenSim screenshot.



OpenSim's Jen Hicks, R&D manager, and Habib discuss plans for the release of OpenSim software.

ment environment (pictured below), is helping researchers and scientists simulate brain activities and simplified brain-like structures that can be used for problem-solving, recognition, prediction, control, modeling, and functional approximations in medicine, robotics, finance, and software.

The Neuroph project started as a university research project at the [University of Belgrade's Faculty of Organizational Sciences](#), in Belgrade, Serbia. It has since evolved into a leading open source project in its field, with contributors from across the globe.



new component as a C++ class, then compile it as a dynamic library that is loaded into the application and its GUI, which is written in Java.

"Cross-platform was key to our selection of Java as the GUI front end to OpenSim," says Habib. "Responsive GUI was a big issue that wouldn't have been possible using other technologies, especially when running computation-intensive tasks. The NetBeans Platform adds modular development, plug-ins, and APIs to support selection, editing, preferences, layouts, docking, and more."

Neuroph, an all-Java neural network framework and integrated develop-



PHOTOGRAPH BY BOB ADLER

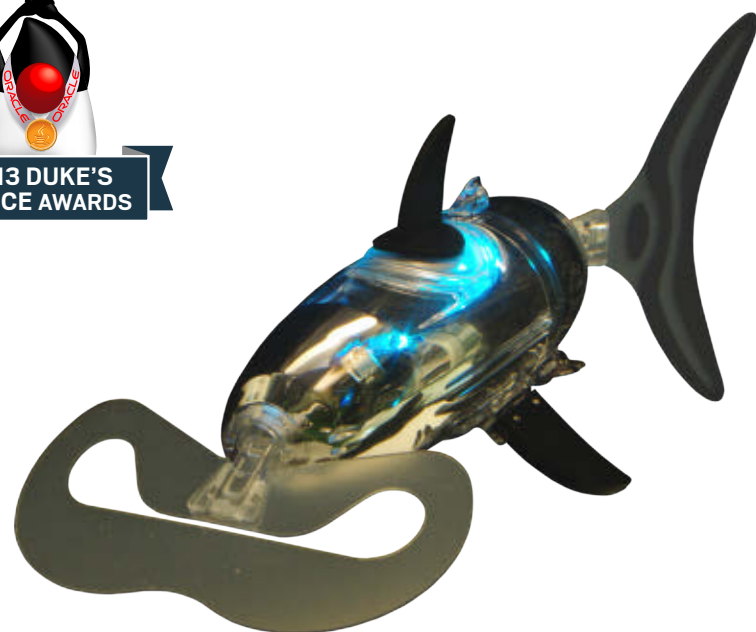
Community Choice 2013 a Tie!

The Java community has spoken, and it's a tie!

This year the two winners of the Community Choice Award are [Bintray](#), a social network for developers from Netanay, Israel—based [JFrog](#), and [Contrast](#), a Java EE security plug-in from Columbia, Maryland—based [Contrast Security](#).

Taking a cue from Facebook, Bintray provides a free, cloud-based social networking platform that enables software developers to download, store, promote, and share executable binary code and libraries. Profile pages list members, available binaries and downloads, relevant websites, bug trackers, community reviews, and the option to watch for future updates.

The Contrast plug-in invisibly monitors applications during testing and automatically identifies security vulnerabilities. Its patented technology weaves "security sensors" into the Java Virtual Machine (JVM), libraries, and the application's custom code, and reports suspected and known vulnerabilities.



PROVIDING GUIDANCE

While OpenSim simulates the human body and Neuroph the human brain, one other winner this year seeks to simulate and analyze something that is at times far more complex and frustrating: traffic. **ISIM**, from intelligent transport system developer [ISBAK](#) of Istanbul, Turkey, is an all-Java traffic planning, simulation, and analysis tool that lets users simulate, plan, and construct road networks using different road, junction, car, speed, and other relevant parameters for maximum results.

In space, the **GEONS Ground Support System (GGSS)** is an analysis and mission operations tool that uses the [GPS-Enhanced Onboard Navigation System \(GEONS\)](#) from the [National Aeronautics and Space Administration \(NASA\)](#). GGSS uses the [NetBeans Platform](#), developed in the [NetBeans IDE](#), as the basis of its ground system software and will support



Left: A Jessiko robotic fish. Right: Robotswim CEO and Founder Christophe Tiraby watches robotic fish with visitors to the RobotWorld conference in Seoul, South Korea.

PHOTOGRAPHS COURTESY OF ROBOTSWIM



JAVA-BASED COMMANDS

“You can use Jessikommand to create movement scenarios and then to send the orders to the robots.”

—Guillaume Genty, Lead Software Engineer, Robotswim



asks them their status, using a patented, two-way optical communication system," explains Guillaume Genty, Robotswim's lead software engineer.

"You can use Jessikommand to create movement scenarios and then to send the orders to the robots, and it also helps robots to stay coordinated with each other," Genty explains. "For example, in some scenarios, we mark a fish as a master and give

it real-time precise orders, and then ask other fish to follow it."

To date, the largest school of Jessiko robotic fish the company has deployed at one time is 50. The typical use is for consumer entertainment, where a school of Jessiko fish are programmed to swim in patterns in aquariums, changing colors as they go. "To create a real school, it is better to have more than 30 robots because some of them are sometimes capricious," Genty adds. "They get lost and continue on their own before going back to the school. It's really artificial life."

Left: A school of robotic fish are powered by a Java-based command and control software. Right: Robotswim's Guillaume Genty and Tiraby take care of the robotic fish environment.

a 2014 launch of the Magnetospheric Multiscale (MMS) mission.

Developed by NASA partner a.i. solutions, the GGSS is deployed in the MMS mission operations control room at the Goddard Space Flight Center, in Greenbelt, Maryland. "Combining JDK 7, the NetBeans Platform, and JavaFX saved an estimated 35 percent in time versus estimates for primary software development," says a.i. solutions Senior Software Engineer Sean Phillips.

Three-year-old French startup Robotswim in Palaiseau, France—just

outside of Paris in what's known as France's Silicon Valley—has produced the world's smallest commercially available robotic fish, Jessiko, measuring 22 centimeters in length. Jessiko is also the only robotic fish that can be programmed to swim in groups, or schools, through the power of **Jessikommand**, a Java-based command and control software. Jessikommand communicates with the fish using a beacon network.

“Jessikommand controls the beacon network and, using those beacons, sends real-time orders to robots and

PHOTOGRAPHS COURTESY OF ROBOTSWIM
AND BY GUILLAUME BONN/GETTY IMAGES



THE FUTURE OF JAVA

Devovx4Kids Founder
Stephan Janssen
(center) problem-
solves with workshop
participants.

JUDGES AND PROCESS

Duke's Choice Award China Winners

Fall 2012 marked an expansion of the Duke's Choice Award program, which now includes regional awards that are announced in conjunction with each international JavaOne conference. One of the highlights of [JavaOne Shanghai](#) was the presentation of the very first Duke's Choice Award China. The 2013 winners were the [Moco technology](#) integration server project; the [X Fantasy](#) MMORPG real-time web-based game; and the [OSChina.NET](#) open source community infrastructure.

Zheng Ye, lead developer of the Moco project, describes Moco as a tool for facilitating the development of applications that interact with web services. "Using web services requires integration,

which involves communications between the different ends," Ye explains. "The consumer of a web service needs to communicate with something to support its development. But what if the server side of the web service is also still under development? Moco is designed to address these issues. Moco enables testing and integration to be completed in a smaller scope, which means simplification."

Jia Ke, lead developer of the X Fantasy game, presented a JavaOne Shanghai session, "[Lessons from Developing the X Fantasy Web Game in Java](#)." When asked what impact he expects Java SE 8 lambda expressions to have on Java game programming, Ke

answered: "What lambda expressions can do is simplify or polish anonymous classes. However, less than one-thousandth of the code in X Fantasy uses anonymous classes, so we don't expect much immediate impact on our

own development. Still, we agree that the support of lambda expressions in Java 8 will bring Java experts more flexibility and choices for optimizing code, which we think is very valuable."

Zhang Hailong, who leads the Guangdong Java User Group, is a cofounder of OSChina—which, with more than a million users, is the largest open source community in China. "The OSChina website is built on top of Java technology—for example, the JDK, Tomcat, and Velocity," Hailong says. "We've expended lots of effort optimizing the website using specific architectures and caching technology. OSChina demonstrates how to build a low-resource-consuming, lightweight, and very fast website using Java technology. We have open-sourced a lot of our code to let other people utilize what we have done." Hailong also notes that OSChina has a forum where Java developers can ask questions and discuss Java technologies. "We also organize offline meetups to enhance the technical atmosphere and help Java developers improve their skills," he says.

—Kevin Farnham



From left: China Duke's Choice winners Jia Ke, Zhang Hailong, and Zheng Ye

PHOTOGRAPHS BY FEIFAN ZHOU/24 EIGA AGENCY
AND COURTESY OF NLJUG



NLJUG's J-Fall conference



"LOCAL FOR LOCAL"

Last year the Duke's Choice Awards established a new tradition by announcing its first-ever awards to a Java user group (JUG)—two, in fact: the London Java Community and JDuchess. This year that tradition continues, with an award going to the [Dutch Java User Group \(NLJUG\)](#)—an organization with national reach throughout the Netherlands. Since its founding in 2004, the organization has grown from about 100 to almost 3,500 members.

"NLJUG considers itself to be a platform from which other initiatives

out of the box. On the `Comparator` class, a `comparing` method takes a function (a lambda) that extracts a comparison key out of the object and returns a `Comparator` that sorts based on that. This means that **Listing 9** could be rewritten even more easily as shown in **Listing 10**.

Think for a moment about what this is doing: the **Person** is no longer about sorting, but just about extracting the key by which the sort should be done. This is a good thing—**Person** shouldn't have to think about how to sort; **Person** should just focus on being a **Person**.

It gets better, though, particularly when we want to compare based on two or more of those values.

Composition. As of Java 8, the **Comparator** interface comes with several methods to combine **Comparator** instances in various ways by stringing them together. For example, the **Comparator.thenComparing()** method takes a **Comparator** to use for comparison after the first one compares. So, re-creating the “last name then age” comparison can now be written in terms of the two **Comparator** instances **LAST** and **AGE**, as shown in **Listing 11**. Or, if you prefer to use methods rather than **Comparator** instances, use the code in **Listing 12**.

By the way, for those who didn't grow up using `Collections.sort()`, there's now a `sort()` method directly on `List`. This is one of the neat things about the introduction of interface default methods: where we used to have to put that kind of noninheritance-based reusable behavior in static methods, now it can be hoisted up into interfaces. (See the [previous article](#) in this series for more details.)

Similarly, if the code needs to sort the collection of `Person` objects by last name and then by first name, no new `Comparator` needs to be written, because this comparison can, again, be made of the two particular atomic comparisons shown in **Listing 13**.

This combinatory “connection” of methods, known as *functional composition*, is common in functional programming and at the heart of why functional programming is as powerful as it is.

It's important to understand that the real benefit here isn't just in the APIs that enable us to do comparisons, but the ability to pass bits of executable code (and then combine them in new and interesting ways) to create opportunities for reuse and design. **Comparator** is just the tip of the iceberg. Lots of things can be made more flexible and powerful, particularly when combining and composing them.

LISTING 7 / LISTING 8 / LISTING 9 / LISTING 10 / LISTING 11 / LISTING 12

```
public static int compareLastAndAge(Person lhs, Person rhs) {
    if (lhs.lastName.equals(rhs.lastName))
        return lhs.age - rhs.age;
    else
        return lhs.lastName.compareTo(rhs.lastName);
}
```



Download all listings in this issue as text

Iteration. As another example of how lambdas and functional approaches change the approach to code, consider one of the fundamental operations done with collections: that of iterating over them. Java 8 will bring to collections a change via the `forEach()` default method defined on the `Iterator` and `Iterable` interfaces. Using it to print each of the items in the collection, for example, requires passing a lambda to the `forEach` method on an `Iterator`, as shown in **Listing 14**.

Officially, the type of lambda being passed in is a **Consumer** instance, defined in the `java.util.function` package. Unlike traditional Java interfaces, however, **Consumer** is one of the new functional interfaces, meaning that direct implementations will likely never happen—instead, the new way to think about it is solely in terms of its single, important method, `accept`, which is the

method the lambda provides. The rest (such as `compose` and `andThen`) are utility methods defined in terms of the important method, and they are designed to support the important method.

For example, `andThen()` chains two `Consumer` instances together, so the first one is called first and the second is called immediately after into a single `Consumer`. This provides useful composition techniques that are a little outside the scope of this article.

Many of the use cases involved in walking through a collection have the purpose of finding items that fit a particular criterion—for example, determining which of the **Person** objects in the collection are of drinking age, because the automated code system needs to send everyone in that collection a beer. This “act upon a thing coming from a group of things” is actually far more widespread than just operating upon a col-

simple math. After all, in any case where a collection of objects can be transformed into a different object (or value) and then collected into a single value, map and reduction operations work.

The map operation, for example, can be useful as an extraction or projection operation to take an object and extract portions of it, such as extracting the last name out of a `Person` object:

```
Stream<String> lastNames =
    people.stream()
        .map(Person::getLastName);
```

Once the last names have been retrieved from the `Person` stream, the reduction can concatenate strings together, such as transforming the last name into a data representation for XML. See **Listing 21**.

And, naturally, if different XML formats are required, different operations can be used to control the contents of each format, supplied either ad hoc, as in **Listing 21**, or from methods defined on other classes, such as from the `Person` class

itself, as shown in **Listing 22**, which can then be used as part of the `map()` operation to transform the stream of `Person` objects into a JSON array of object elements, as shown in **Listing 23**.

The ternary operation in the middle of the `reduce` operation is there to avoid putting a comma in front of the first `Person` serialized to JSON. Some JSON parsers might accept this format, but that is not guaranteed, and it looks ugly to have it there.

It is ugly enough, in fact, to fix. The code is actually a lot easier

to write if we use the built-in `Collector` interface and its partner `Collectors`, which specifically do this kind of mutable-reduction operation (see **Listing 24**). This has the added benefit of being much faster than the versions using the explicit `reduce` and `String::concat` from the earlier examples, so it's generally a better bet.

Oh, and lest we forget our old friend `Comparator`, note that `Stream` also has an operation to sort a stream in-flight, so the sorted JSON represen-

BE A COLLECTOR

It is ugly enough to fix. The code is actually a lot easier to write if we use the built-in `Collector` interface and its partner `Collectors`, which specifically do this kind of mutable-reduction operation.

LISTING 21 LISTING 22 LISTING 23 LISTING 24 LISTING 25 LISTING 26

```
String xml =
    "<people data='lastname'>" +
    people.stream()
        .map(it -> "<person>" + it.getLastName() +
            "</person>")
        .reduce("", String::concat)
    + "</people>";
System.out.println(xml);
```



[Download all listings in this issue as text](#)

tation of the `Person` list looks like **Listing 25**.

This is powerful stuff.

Parallelization. What's even more powerful is that these operations are entirely independent of the logic necessary to pull each object through the `Stream` and act on each one, which means that the traditional `for` loop will break down when attempting to iterate, map, or reduce a large collection by breaking the collection into segments that will each be processed by a separate thread.

The `Stream` API, however, already has that covered, making the XML or JSON `map()` and `reduce()` operations shown earlier a slightly different operation—instead of calling `stream()` to obtain a `Stream` from the collection, use `parallelStream()` instead,

as demonstrated in **Listing 26**.

For a collection of at least a dozen items, at least on my laptop, two threads are used to process the collection: the thread named `main`, which is the traditional one used to invoke the `main()` method of a Java class, and another thread named `ForkJoinPool.commonPool.worker-1`, which is obviously not of our creation.

Obviously, for a collection of a dozen items, this would be hideously unnecessary, but for several hundred or more, this would be the difference between “good enough” and “needs to go faster.” Without these new methods and approaches, you would be staring at some significant code and algorithmic study. With them, you can write parallelized code literally by adding eight keystrokes

Web applications in pure Java?

For over a decade, the web and internet have been the platform for all kinds of applications – personal and business. The biggest benefit of a web application has been the easiness to reach the audience's desktops without software installations, with a single deployment.

But at the same time we have been pushing the limits of browser technologies, most importantly HTML and JavaScript, to create even better user interfaces that meet the expectations.

The key elements in the architecture of a web based system are of course the browser and the application server.

There has been a long lasting fight over which side the applications should run on. In the browser you can have better responsiveness, interactivity and visuality for the

“Vaadin introduces a familiar user interface architecture that is based on Java classes, components and widgets.”

user, but server-side environment is much easier to manage – mostly because of solid component and service oriented architectures like Java EE. But what we want to have is best of both worlds.

Vaadin is an open-source web application architecture that brings more balance to this server-client fight or dilemma. Vaadin introduces a familiar user interface architecture that is based on Java classes, components and widgets. It is very quick to learn.

The key feature is that widgets functionally span from server to client. Think of a 3D animated coverflow widget where the data is programmatically and securely bound in the server, but we have these fancy scroll effects using the latest browser technology. All this seamlessly in a single logical component.

Heavily based on object oriented Java language and dynamic loading and management of runtime components in Java EE, Vaadin brings you the reusability, expressiveness and programmatic easiness you already saw in old Java Swing applications. This time for pure web applications. In pure Java language.

LEARN MORE

- [JavaOne 2013 booth #5106](#)
- [The Vaadin open source project: vaadin.com](#)
- [Live demo: vaadin.com/demo](#)

The #1 conference for Java web applications.

Join us for GWT.create in December – **Register now at gwtcreate.com**

GWT.create^{<US/EU>}

San Francisco
California

December 12–13th 2013

Frankfurt am Main
Germany

December 17–18th 2013

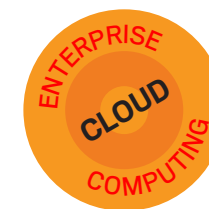
CLOUD: CHALLENGE AND OPPORTUNITY

Cameron Purdy on cloud development

BY TIMOTHY BENEKE

ART BY I-HUA CHEN; PHOTOGRAPHY BY BOB ADLER

Oracle Vice President of Development
Cameron Purdy (left) and Architect Rob Lee
talk about Oracle Coherence features.



According to Cameron Purdy, vice president of development at Oracle, the advent of cloud computing has led to a new wave of software innovation comparable to the surge in web-oriented technologies 15 years ago or the rise of business applications and the Java EE platform some 10 years ago. The potential cost savings and enhanced productivity offered by cloud computing present Java developers with new challenges and opportunities.

To meet these challenges, Java developers and architects will need to change the way they think. Purdy, who has worked with Java technology since 1996, is in a unique position to address these changes. Prior to joining Oracle in 2007, he was the CEO of Tangosol, whose Coherence



Left to right: Team members Lee, Nilesh Junnarkar, and Dhiraj Mutreja discuss clustering and cloud services with Purdy.

Data Grid product offered reliable and scalable data management across the enterprise. He regularly participates in industry standards development and is a specification lead for the Java Community Process (JCP). He has five times been named a JavaOne Rock Star for the quality of his JavaOne presentations, and he was recognized in TheServerSide's "Who's Who in the Enterprise Java World."

Java Magazine talked with Purdy to hear his latest thinking on cloud computing, Java, and Oracle's new cloud foundation products.

Java Magazine: Why should Java developers pay attention to cloud computing?

Purdy: First, it's important to appreciate the wide scope of the term *cloud computing*. Obviously, we talk about

the public cloud, including Amazon EC2 or Microsoft Azure, not to mention Oracle's own public cloud. But cloud computing is also changing the way that we manage internal data centers—what we refer to as the *private cloud*. In other words, the very same concepts, efficiencies, and capabilities that the public cloud brought us are now available inside the data center—for example, basic tenets such as self-service and being able to get a development, staging, or production environment for an application in minutes—without going through any significant paperwork or human workflow. On top of that, the elasticity of the cloud means that we're no longer constrained by a one-size-fits-all model.

Java Magazine: What do Java developers need to understand about this?

Purdy: First, applications must be built so that they can dynamically scale out and dynamically balance the load across multiple servers. This means that when servers are added while an application is running, there's no interruption of service. To accomplish this, it's good programming practice to always assume that an application or the components of an application are running in a scaled environment, which

means many previous assumptions no longer apply. I'm talking about things such as file systems—which could be either local or shared and could trip up developers either way—or global Java objects on the heap, which won't actually be “global” when the application is scaled out.

Developers must also understand whether the addition of resources will actually enable an application to scale out effectively. In other words, just because an application runs correctly when you add additional servers, that doesn't mean it will actually support more transactions or users. So, you need to explicitly design into the scale-out model the ability for the application to scale as close to linearly as possible. Obviously, data caching is a huge part of this, as is minimizing the amount of information that has to be

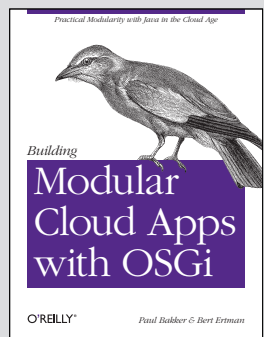
shared across servers, as is minimizing contention on shared resources such as database systems.

Finally, the application needs to be understood in terms of the metrics of elasticity: when are additional resources actually needed, and when is it safe for resources to be taken away? So, you need to know *when* you need to scale out the application, which involves knowing what to monitor.

ELASTIC CLOUD

We're no longer constrained by a one-size-fits-all model. Applications must be built so that they can dynamically scale out and dynamically balance the load across multiple servers.

How Developers Are Using the Cloud



We talked with Bert Ertman, Java Champion and Netherlands JUG leader, about his experience as a developer with the cloud.

Java Magazine: How are you using the cloud?

Ertman: I'm using the cloud as a development and deployment platform for realizing sophisticated software-as-a-service [SaaS] offerings for my customers.

Java Magazine: What are the benefits?

Ertman: The cloud enables pay-as-you-go, both for development and testing and also for running costs. Using a stateless architecture and leveraging the advantages of a modular application architecture, we are able to achieve horizontal scalability and auto-scaling. This way, we pay only for what we truly need instead of wasting vast amounts of money on hardware and data centers.

Java Magazine: How has the cloud changed your job?

Ertman: As a developer, you have to be aware of the underlying infrastructural mechanics. From this perspective, DevOps is an interesting trend. Also, the cloud challenges a lot of the traditional development stack: relational database versus NoSQL, proprietary interfaces versus open APIs using RESTful services, stateful versus stateless, transactional versus eventual consistent, server-side web frameworks versus HTML5 and MVC JavaScript, and so on. Building cloud applications is more than just building a web app and deploying it on virtual hardware.

Java Magazine: Are you willing to share any of your hard-won knowledge about building apps for the cloud?

Ertman: Absolutely! My coworker Paul Bakker and I have just finished writing a book for O'Reilly called *Building Modular Cloud Applications with OSGi*. The book is a practical guide to applying the concepts of modularity to the tools and technologies that come as part of the cloud development stack. The book leverages what we have learned so far in the past couple of years. Right now the book is in production, and we hope to have it available just ahead of JavaOne San Francisco.

You can find Bert speaking at JavaOne San Francisco and at the [J-Fall conference](#) in the Netherlands, and online [@BertErtman](#).

—Tori Wieldt, Senior Java Community Manager, Oracle

Java Magazine: How does Java EE 7 enable these kinds of scaling?

Purdy: Several improvements in Java EE 7 are relevant. Obviously, the new batch specification, [JSR 352](#), is tremendously valuable, and not only for cloud computing. It's able to take significant amounts of work that might otherwise have to be done synchronously and break it down into smaller units and perform those asynchronously. In addition, the Java EE role definitions have been updated to better map to the security requirements we see in cloud environments, such as the difference between someone administering platform as a service [PaaS], on which the application is running, versus administering the application itself that is being hosted.

Software also works better in a cloud environment when we get rid of certain assumptions. One way to do this is through dependency injection, which was introduced in Java EE 5 and, as of Java EE 7, now applies across the entire Java set of enterprise specifications and is called Context and Dependency Injection [CDI]. Instead of going out and finding what you need as a component, you simply declare what you need. If you need to connect to a database, the component declares that it needs to be connected to a

database and then it allows whatever environment it's running in to provide it with a connection to that database based on how that environment is configured.

Java Magazine: Could you clarify infrastructure as a service [IaaS] and PaaS with respect to Java EE?

Purdy: IaaS manages the requisite networking and server hardware, and it typically hosts virtual machines, each of which is running an operating system—all of which is below the level of Java EE. We imagine that every data center, every private cloud, and every public cloud will be providing IaaS: infrastructure services that platforms and applications can take advantage of through standardized IaaS APIs.

Generally speaking, what developers are looking for is PaaS. Developers need to be able to describe an application hosting environment: an application server or a cluster of application servers, which are often combined with a database. And they need to have their application exposed to the real world, protected by a firewall, and automatically balanced by a load balancer.

In other words, they're not interested in building or installing their own firewall, load balancer, or database. They want to be able to take an application, including necessary components and their database design, and deploy it in a way that is



Purdy fuels up for the day with coffee and a banana.

and tablets. And part of that is certainly its HTML5 support capability. And with many of these applications, time to market—being able to build and deploy applications quickly and at low cost, and being able to scale those applications out to handle growth—is essential. Many of the applications that we've been talking about target mobile platforms, often by providing RESTful services. And these are all areas that are targeted by Java EE 7: support for HTML5, RESTful web services, WebSockets, and JSON. These applications are typically exposing RESTful services that return JSON data, and Java EE 7 has great support for building and parsing JSON data.

Java EE 7 has much better support for providing RESTful services as well, including support for asynchronous RESTful services.

It seems that every generation of software expands its complexity to the point where it's no longer manageable. The reason the cloud is emerging as a dominant technology movement is because we need automation of management and simplification of infrastructure and platform just to manage the complexity that is inherent in today's applications. Similarly,

HTML5 is needed in order to provide the level of interactivity and the richness of applications that people have been trying to handcraft and also to work around the limitations and complexity of supporting various proprietary client technologies.

Java Magazine: How does Oracle WebLogic Server 12.1.2 fit into the cloud?

Purdy: Oracle WebLogic Server 12.1.2, which is part of Oracle Cloud Application Foundation, contains a number of exciting new capabilities, such as sup-

port for RESTful services and HTML5 WebSockets. One of the features we're most excited about is what we call *dynamic clusters*. Dynamic clusters allow an application to dynamically add servers on the fly. This used to have to be manually preconfigured and now it's automated, which offers a very elastic environment for the applications we've been discussing. Also, Oracle WebLogic Server 12.1.2 supports the new Oracle Database 12c release, including its multitenancy option, which is a significant capability for developing cloud applications.

Java Magazine: What is *multitenancy*, and why is it important?

Purdy: Assuming that an application is providing software as a service [SaaS], multitenancy means that each organization that's purchasing that service has its own secure set of data.

Typically, it would be very expensive from an infrastructure or management point of view to give each tenant its own database, but a true multitenanted database capability is actually built into Oracle Database 12c. Oracle WebLogic Server works very closely with the database to take advantage of that capability, including taking advantage of what's called *Database Resident Connection Pooling*

IT'S COMPLICATED
We need automation
of management
and simplification
of infrastructure
and platform just
to manage the
complexity that is
inherent in today's
applications.

article can be downloaded [here](#).

It should be noted that this demo application is by no means a professional chat application. The goal is to explain some of the core aspects of WebSocket support in Java, rather than to create a secure, robust, and feature-complete chat environment.

The server component. The server component of the chat application is very lightweight. A list of active chatters is maintained, but apart from that, no state is preserved. The server component listens for incoming messages on WebSockets, and sends messages to the client over the same WebSockets. The client implementations create a WebSocket connection to the server and exchange messages. The WebSocket specification is very open regarding the type of content that can be transmitted. For this demo application, we use the JSON format for encoding messages, because JSON is widely supported in Java and JavaScript.

Our demo application supports the following scenario:

- An end user visits a web page or starts a Java application.
- The web page or application opens a WebSocket connection to the server.
- The server immediately sends a list of active chatters to the client.

- Simultaneously, the end user is allowed to enter his nickname.
- The nickname is sent to the server.
- The server sends a login message with the nickname to all active clients.
- The clients process the login message and add the new user to the list of active chatters.
- The end user can then enter a new chat message, which is sent to the server.
- The server distributes the new chat message to all active clients.
- When the end user leaves the chat, the client WebSocket connection to the server is closed.
- The server detects the closed connection, and sends a log-out message to the remaining clients.
- The clients process the logout message and remove the specific user from the list of active chatters.

The server component contains the following parts:

- **ChatEndpoint** is the class that contains the logic implementing the lifecycle methods related to the WebSocket operations.
- **ChatCommand** and its subclasses contain specific commands and information that are exchanged between the server and the clients.
- **ChatDecoderEncoder** is a util-

LISTING 1

LISTING 2

```
@ServerEndpoint(value="/endpoint", decoders=ChatDecoderEncoder.  
class, encoders=ChatDecoderEncoder.class)  
public class ChatEndpoint {  
...  
}
```



[Download all listings in this issue as text](#)

ity class that translates **ChatCommand** instances to JSON strings and vice versa.

`ChatCommand` and its subclasses define the application protocol between server and clients. All `ChatCommand` instances have a field named `command` that defines the type of the command (`login`, `logout`, `allusers`, or `message`). The clients will send and receive JSON representations of these commands and can, thus, also distinguish between the different types of commands.

The `ChatEndpoint` class is annotated with the `@ServerEndpoint` annotation as shown in **Listing 1**.

We declare three elements in the `@ServerEndpoint` annotation:

- `value="/endpoint"` specifies that we want to listen for incoming WebSocket connections at the URL specified by the context root of this application plus `"/endpoint"`. When a client connects a WebSocket to this address, an instance of the `ChatEndpoint`

class will be able to react to lifecycle events (for example, `@OnOpen`, `@OnMessage`, `@OnClose`, and `@OnError`).

- `decoders=ChatDecoderEncoder`
`.class` specifies that the `ChatDecoderEncoder` utility class will be used for decoding incoming text messages into `ChatCommand` instances.
- `encoders=ChatDecoderEncoder`
`.class` specifies that the `ChatDecoderEncoder` utility class will be used for encoding `ChatCommand` instances into (JSON) text messages.

We want to maintain a list of active users. For this simple application, we will use the static `Map` shown in **Listing 2**. Each entry in the `sessionUsers` map contains a `WebSocket` session and the name of the user associated with the specific `WebSocket` session.

When a client establishes a WebSocket connection to the URL specified in the `value` element of the `@ServerEndpoint` annota-

tion, the method annotated with `@OnOpen` will be called. This method might contain a number of parameters: the `Session` instance, an `EndpointConfig` parameter, and a number of `PathParam` parameters corresponding to path information. In our example, we need to know only the `Session` instance, which is created by the underlying implementation and which uniquely identifies the associated WebSocket (including the client peer that initiated the connection).

Our implementation of the `@OnOpen` method has two goals:

- Send the list of active chatters to the client.
- Register the new WebSocket session with our application. Until we know the name of the user who will chat, we associate an anonymous user with this session.

These goals are achieved using the code shown in **Listing 3**.

The list of active chatters is obtained by taking the values of the `sessionUsers` map. We create an instance of the `AllUsersCommand` class and populate it with the list of active chatters. The instance is then sent to the remote peer using the `Session.getBasicRemote().sendObject(...)` method.

Because the `ServerEndpoint` is configured with the

The `ChatDecoderEncoder` class as the encoder for messages, the `AllUsersCommand` instance is converted to a `String` object. The WebSocket implementation will, therefore, call `ChatDecoderEncoder.encode(ChatCommand object)`, which returns a `String` object. The resulting `String` object is then sent over the open WebSocket to the client, where it can be parsed.

The conversion between the instances of `ChatCommand` and JSON strings is achieved using [JSR 353](#), the JSON API for Java EE.

The implementation of this method is beyond the scope of this article, and can be found in **Listing 4**.

When a user wants to participate in the chat, the client sends a login message containing the nickname of the user. We assume that the client sends a JSON string containing the required information.

When our [ServerEndpoint](#) receives a message, the method annotated with [@OnMessage](#) is called. In our demo application, this method has the signature shown in **Listing 5**. The method annotated with [@OnMessage](#) can have a number of optional parameters, as explained in the [Javadoc](#) for [@OnMessage](#).

In our case, the method will be called with an instance of `ChatCommand` and the `Session`

LISTING 3

LISTING 4

LISTING 5

```
@OnOpen
public void onOpen (Session s) {
    AllUsersCommand auc = new AllUsersCommand();
    auc.setUids(sessionUsers.values());
    try {
        s.getBasicRemote().sendObject(auc);
    } catch (IOException | EncodeException ex) {
        Logger.getLogger(ChatEndpoint.class.getName()).log(Level.SEVERE, null, ex);
    }
    sessionUsers.put(s, "anonymous");
}
```



[Download all listings in this issue as text](#)

parameter that uniquely defines the WebSocket session. The WebSocket implementation will create the `ChatCommand` instance by calling the `ChatEncoderDecoder.encode(String)` method, converting the JSON string sent by the client peer of the WebSocket into an instance of `ChatCommand`.

This is a very convenient approach, because it allows a clean separation between the content of the message (contained in the `ChatCommand` class) and the transport protocol (JSON). The `ChatEndpoint` instance does not need to be aware of the transport protocol. By changing the `encoders`

element in the `@ServerEndpoint` annotation, a different transport protocol can be used.

The implementation of our method annotated with `@OnMessage` will do the following:

- In case the supplied **ChatCommand** is a **LoginCommand**, the nickname of the user will replace the “anonymous” value that initially was associated with the WebSocket session.
- The message will be multicasted to all active chatters.

These goals are achieved with the code shown in **Listing 6**.

Note that we send the message to all active chatters by iterating over the `keySet` of the `sessionUsers` map. This requires that we keep track of connections that are broken and users who are leaving.

We assume that the clients will close the WebSocket connection when the active user is leaving (either by closing the application or by pressing a logout button). Good enough; the Java WebSocket API has the `@OnClose` annotation, which can be used to indicate

STAY TUNED

In this article, we create an HTML5 client. In Part 2, we will create a Java-based client, leveraging the JSR 356 client API. The server and the two clients use the same underlying protocol, so they work together seamlessly.

that a certain method should be called when a WebSocket session is closing. The underlying WebSocket implementation will make sure that the method annotated with `@OnClose` will be called if a WebSocket connection is closed (either intentionally or because of network issues).

In our case, the `@OnClose` annotated method will send a logout command to all other sessions and subsequently remove this session from the

list of active chatters. This behavior is obtained using the code in **Listing 7**.

Note: We could also use the convenient `session.getOpenSessions()` method to obtain a list of active sessions.

HTML5 client. A very simple version of an HTML5 client that connects to the server component described above is shown in the file in the source code called `client/html5/chat.html`.

When a user enters the web page, a WebSocket to the server URL will be created. We define a `body onLoad` handler that contains

LISTING 6 LISTING 7 LISTING 8

```
@OnMessage
public void onMessage (ChatCommand cmd, Session s) throws IOException,
    EncodeException {
    if (cmd instanceof LoginCommand) {
        String uid = ((LoginCommand)cmd).getUid();
        sessionUsers.put(s, uid); // this will overwrite the "anonymous" user
    }
    for (Session session : sessionUsers.keySet()) {
        session.getBasicRemote().sendObject(cmd);
    }
}
```

 [Download all listings in this issue as text](#)

the required functionality to create a WebSocket:

```
<body onLoad="openSocket();">
...
</body>
```

The `openSocket()` function is defined in **Listing 8**.

The WebSocket interface in **Listing 8** is defined as part of the W3C draft of the [WebSocket API](#), and it is available in most modern

browsers. The WebSocket interface allows developers to provide the following lifecycle callback functions:

- onopen
- onmessage
- onerror
- onclose

Note that these functions are conceptually similar to the life-cycle annotations in the Java API for WebSocket (`@OnOpen`, `@OnMessage`, `@OnError`, and

@OnClose). This makes the development of an end-to-end, bidirectional WebSocket-based application much easier.

Because our server component will immediately send a list of active chatters upon opening a WebSocket connection, the `.onmessage` handler on the `connection` object will be called. In this handler, we receive the message coming from the server in JSON format, which can easily be processed in JavaScript.

The code in **Listing 9** first detects the type of the incoming message. This can be achieved by inspecting the `command` field of the incoming JSON data. The value of this field corresponds to the value of the `ChatCommand` `.getCommand()` method in the server component.

If the incoming message is an `allusers` command, the function will populate an HTML element with the nicknames of the active users (see **Listing 10**).

The `AllUsersCommand` contains a field named `uids`, which contains

the list of active chatters. We can easily obtain this list in JavaScript (`allusers = x.uids`) and iterate over the active users. For each user, we create a new HTML paragraph and we set the `innerHTML` of that paragraph to the user ID (nickname) of the user.

We also add a unique identifier to the element, because we should be able to retrieve and remove the element when that

specific user leaves the chat. In that case, the client will receive a `logout` command from the server. The incoming message is processed as shown in **Listing 11**.

As a consequence, the name of the user who left is no longer shown. Note that in a more powerful chat application, users who have left the chat could be shown in a different style.

When a new user enters the chat while our client is active,

we will be notified via a `login` command sent by the server. In that case, we retrieve the nickname (or user ID) for the user, and add it to the `chatters` element using the code in **Listing 12**.

WEBSOCKET SPEC

The W3C has a working draft describing **how to leverage WebSockets from a web page**, and ever since Java EE 7, a similar specification has existed for dealing with WebSockets in Java. This specification is defined in JSR 356.

LISTING 9 / LISTING 10 / LISTING 11 / LISTING 12 / LISTING 13 / LISTING 14

```
var x = JSON.parse(evt.data);
command = x.command;
if (command == "allusers") {
    ...
}
if (command == "login") {
    ...
}if (command == "logout") {
    ...
}if (command == "message") {
    ...
}
}
```



Download all listings in this issue as text

The web page contains a text field for entering the user's nickname and a button for joining the chat. Clicking the button will call the `loginUser()` function, which will send a login message to the server. Also, the list of active chat-
ters (which was already retrieved when starting the WebSocket connection) will be shown, as well as a chat box containing the messages. All of this is achieved with the function shown in **Listing 13**.

The code in **Listing 13** shows how easy it is to send a message over a WebSocket connection to a server component. Using the `JSON.stringify` function, a JSON message is created and sent to the

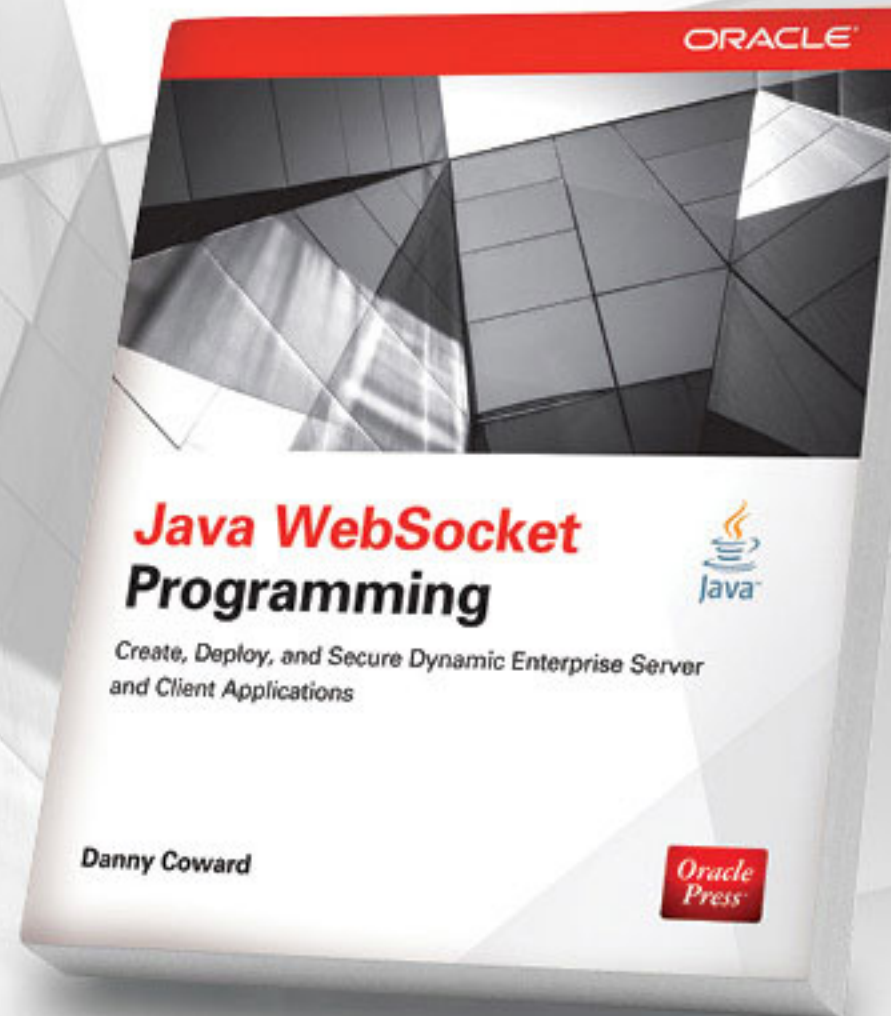
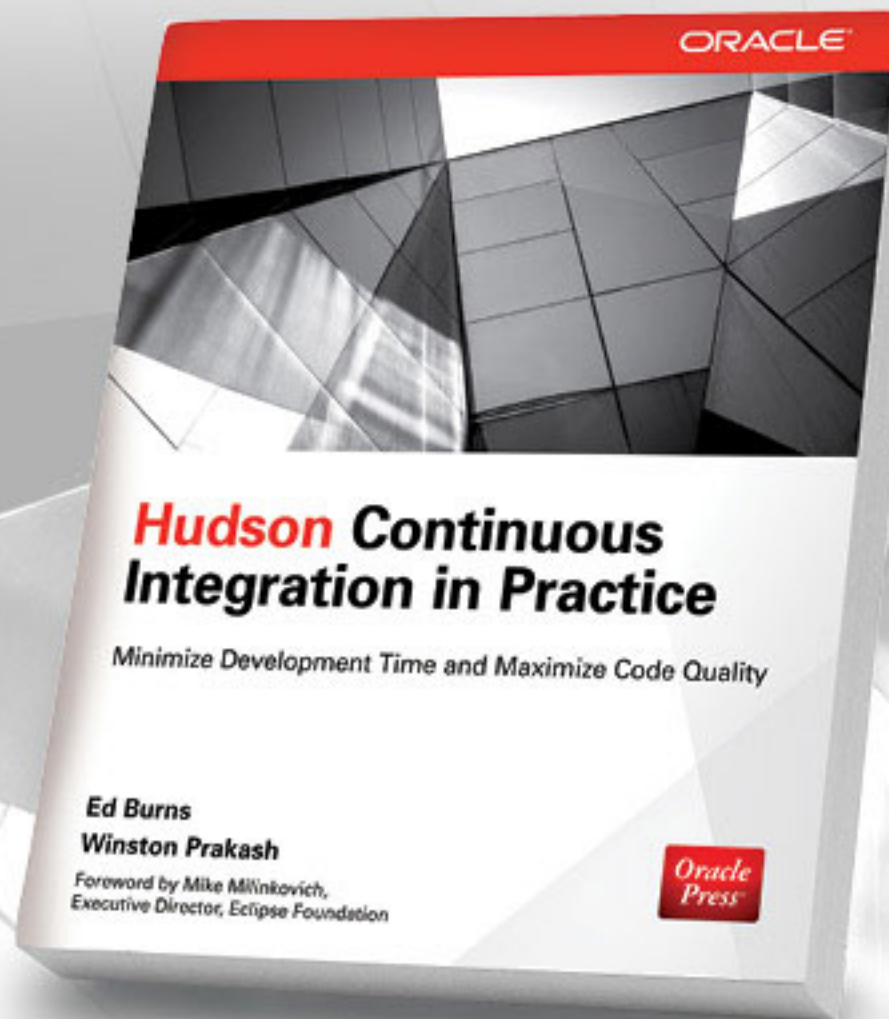
server, where the JSON message is decoded into a Java object (an instance of `ChatCommand`).

When the user enters a chat message, the `talk` function is called, and the content of the message (contained in a text field named `msg`) is sent to the server, as shown in **Listing 14**.

When a user sends a message, all clients will receive a `message` command. Our client will add the content of that message to the `messages` HTML element. **Listing 15** shows how this is achieved.

While this example is far from complete, it shows how an HTML5 client can easily communicate with a Java back end and exchange

Written by leading technology professionals, Oracle Press books offer the most definitive, complete, and up-to-date coverage of Java available.



Hudson Continuous Integration in Practice

Ed Burns and Winston Prakash

Streamline each process in your development lifecycle to optimize productivity while reducing risk and complexity.

Java WebSocket Programming

Danny Coward

The top expert on Java WebSocket programming shows how to build dynamic enterprise Web applications that fully leverage state-of-the-art communication technologies.

**Oracle
Press™**



JavaFX Data Binding for Enterprise Applications

JAVA IN ACTION

passes the current time to the listeners. The Pomodoro technique is sometimes used by programmers to

Java SE does not support data binding at the language level. You cannot just con-

JAVA TECH

ABOUT US



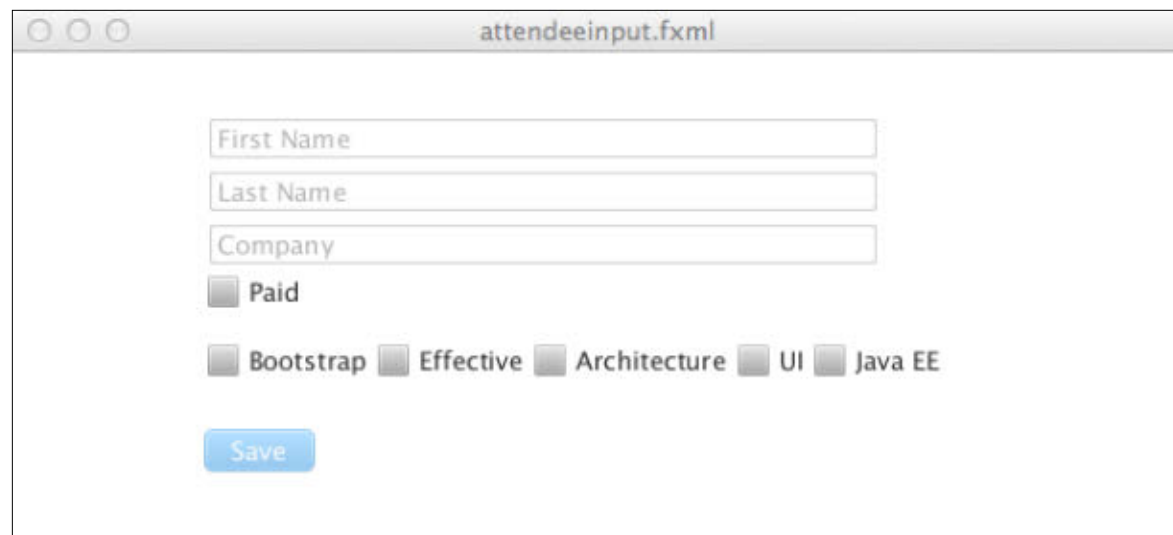


Figure 1

sustain long documentation writing periods.

Pomodoro extends the abstract `ObjectBinding` class, which significantly simplifies the implementation. Only the method `computeValue` needs to be implemented. All the `ChangeListener` bookkeeping is already implemented by the `ObjectBinding`.

Every second, the internal timer fires (in the real world, the length of the period is 25 minutes with a five-minute break) and invokes the `invalidate` method. Behind the scenes, all registered listeners and observers are notified and synchronized. The code in **Listing 5** demonstrates the usage.

All JavaFX UI components also expose a set of bindable properties, which can be used for direct synchronization with presenters and models.

A Binding DSL

JavaFX offers a higher-level binding API, which allows you to perform computations on the fly. Instead of just synchronizing the plain values of two bound objects, additional computations can be performed in real time, as shown in **Listing 6**.

The higher-level API supports not only numerical operations but also **String** and **Boolean** properties, as shown in **Listing 7**. String binding is particularly useful for the implementation of bindable “live templates” within the application, such as status bars or labels for UI components.

Validating the Input

The binding DSL is particularly useful for input validation. Imagine an input view that allows you to save only coherent data

LISTING 1

```
import javafx.beans.property.SimpleStringProperty;
import javafx.beans.property.StringProperty;
import static org.junit.Assert.*;
import static org.hamcrest.CoreMatchers.is;
import org.junit.Test;
```

```
@Test
public void bindString() {
    StringProperty input = new SimpleStringProperty();
    StringProperty output = new SimpleStringProperty();
    output.bind(input);
    input.set("duke");
    assertThat(output.get(), is("duke"));
}
```

 [Download all listings in this issue as text](#)

(see **Figure 1**). The First Name, Last Name, and Company fields need to be filled in and one of the workshop name checkboxes has to be selected in order to make the

Save button active. Traditionally, such validation was implemented with a series of **if-else** statements, sometimes factored out into a **Mediator**. **Listings 8a** and **8b** show

The `ObjectProperty` selected `Attendee` is changed by some other presenter and contains the currently active domain object. The `AttendeeInputPresenter` subscribes itself as a listener to this property

WHAT A SURPRISE

Data binding **has surprising effects** on application design. With JavaFX presenters, capabilities can be cleanly exposed as properties.

in the `initialize` method
and reacts to the
changes.

On the other side, the `newAttendeeProperty` is the outbound communication channel. Every time a new attendee is created by the user, the declaratively bound `save` method is executed and the state of the `newAttendeeProperty` is changed. Any other interested presenter or view can react to the changes without any coupling or knowledge.

In fact, the application-level presenter, `AirhacksPresenter`, wires all the independent presenters together in the initialization phase, as shown in **Listing 11**.

The state of the `selectedAttendeeProperty` did not require any further transformations or filtering, so it was directly bound. Additional filtering is performed before passing the value in the anonymous `ChangeListener` implementation, so the `AttendeeInputPresenter` and the `WorkshopPresenter` were glued together with a `ChangeListener`.

Without the null check, you could also directly bind both properties. Similarly, the `WorkshopPresenter` consumes the `selectedAttendeeProperty` and offers the `new`

LISTING 10a / LISTING 10b / LISTING 11 / LISTING 12a / LISTING 12b

```
package com.airhacks.control.presentation.attendeeinput;44444
import
com.airhacks.control.business.registrations.entity.Attendee;
import javafx.beans.property.SimpleBooleanProperty;
import javafx.beans.property.SimpleObjectProperty;
import javafx.beans.property.ObjectProperty;
import javafx.beans.value.ChangeListener;
import javafx.beans.value.ObservableValue;
import javafx.fxml.Initializable;

public class AttendeeInputPresenter implements Initializable {
    private ObjectProperty<Attendee> selectedAttendee;
    private ObjectProperty<Attendee> newAttendee;

    @Override
    public void initialize(URL url, ResourceBundle rb) {
        this.selectedAttendee = new SimpleObjectProperty<>();
        this.newAttendee = new SimpleObjectProperty<>();
        this.selectedAttendee.addListener(
            new ChangeListener<Attendee>() {
                @Override
                public void changed(
                    ObservableValue<? extends Attendee> observable,
                    Attendee oldValue, Attendee newValue) {
                        //
                    }
                });
    }
}
```

 [Download all listings in this issue as text](#)

AttendeeProperty to others at the same time. The **WorkshopPresenter** also acts as a dispatcher and coordinates subordinate **DayPresenter** instances with cor-

responding views. There is still no direct coupling required between the `WorkshopPresenter` and the `AttendeeInputPresenter` (see **Listings 12a** and **12b**).



In the July/August issue, Simon Ritter gave us a code challenge around JavaFX. He presented us with a JavaFX application that tries to place a button on each side of the divider in a SplitPane.

The correct answer is #3. Each button needs to be placed in its own Pane (StackPane, FlowPane, or any other subclasses would also work).

Now when the application is run, the divider is in the center of the window, with the buttons positioned in the top left corner of each half of the SplitPane.

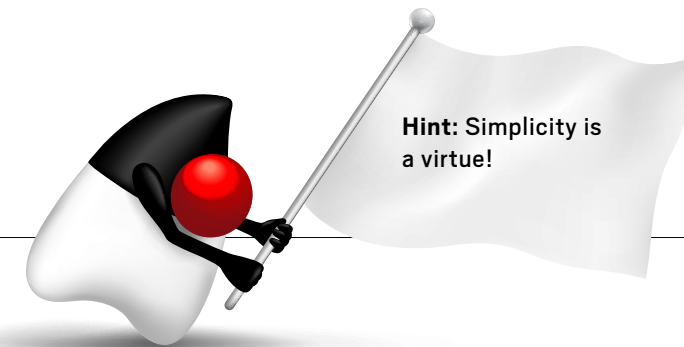
This issue's challenge comes from Attila Balazs, a polyglot developer from Cluj-Napoca, Romania, who presents us with a class initialization challenge.

1 THE PROBLEM

A developer in your group struggles with the following code. Most of the time it deadlocks, and sometimes it throws a `NullPointerException` but it never runs to the end correctly. How can you fix the problem?

2 THE CODE

```
final class App {
    public static final Integer RETRIES = 3;
    private static final App instance = new App(Controller
        .getInstance());
}
```



Hint: Simplicity is a virtue!

```
private Controller ctrl;
```

```
public App(Controller controller) { this.ctrl = controller; }
public void run() { ctrl.run(); }
public static App getInstance() { return instance; }
}
```

```
final class Controller {
    private static final Controller instance = new Controller(App
.RETRIES);
    private final Integer retries;
```

```
private Controller(Integer retries) { this.retries = retries; }
public static Controller getInstance() { return instance; }
public void run() { }
public Integer getRetries() { return retries; }
}
```

// ...

```
// some thread starts running the app
new Thread(new Runnable() {
    public void run() { App.getInstance().run(); }
}, "AppThread").start();
```

```
// meanwhile, on the main thread
Controller.getInstance().getRetries();
```

3 WHAT'S THE FIX?

- 1) Use int instead of Integer for RETRIES.
- 2) Use jstack to find the deadlock.
- 3) Move RETRIES into Controller.
- 4) Use -XX:CompileThreshold=0 to inline App.RETRIES

GOT THE ANSWER?

Look for the answer in the next issue. Or submit your own code challenge!